

Probabilistic Anonymity and Admissible Schedulers

Flavio D. Garcia, Peter van Rossum, and Ana Sokolova

Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands.
<http://www.cs.ru.nl/~{flaviog, petervr, anas}>

Abstract. When studying safety properties of (formal) protocol models, it is customary to view the scheduler as an adversary: an entity trying to falsify the safety property. We show that in the context of security protocols, and in particular of anonymizing protocols, this gives the adversary too much power; for instance, the contents of encrypted messages and internal computations by the parties should be considered invisible to the adversary.

We restrict the class of schedulers to a class of admissible schedulers which better model adversarial behaviour. These admissible schedulers base their decision solely on the past behaviour of the system that is visible to the adversary.

Using this, we propose a definition of anonymity: for all admissible schedulers the identity of the users and the observations of the adversary are independent stochastic variables. We also develop a proof technique for typical cases that can be used to prove anonymity: a system is anonymous if it is possible to ‘exchange’ the behaviour of two users without the adversary ‘noticing’.

1 Introduction

Systems that include probabilities and nondeterminism are very convenient for modelling probabilistic (security) protocols. Nondeterminism is highly desirable feature for modelling implementation freedom, action of the environment, or incomplete knowledge about the system.

It is often of use to analyze probabilistic properties of such systems as for example “in 30% of the cases sending a message is followed by receiving a message” or “the system terminates successfully with probability at least 0.9”. Probabilistic anonymity [BP05] is also such a property. In order to be able to consider such probabilistic properties we must first eliminate the nondeterminism present in the models. This is usually done by entities called schedulers or adversaries. It is common in the analysis of probabilistic systems to say that a model with nondeterminism and probability satisfies a probabilistic property if and only if it satisfies it no matter in which way the nondeterminism was resolved, i.e., for *all possible schedulers*.

On the other hand, in security protocols, adversaries or schedulers are malicious entities that try to break the security of the protocol. Therefore, allowing just any scheduler is inadmissible. We show that the well-known Chaum’s Dining Cryptographers (DC) protocol [Cha88] is not anonymous if we allow for all possible schedulers. Since the protocol is well-known to be anonymous, this shows that for the treatment of probabilistic security properties, in particular probabilistic anonymity, the general approach to analyzing probabilistic systems does not directly fit.

We propose a solution based on restricting the class of all schedulers to a smaller class of *admissible schedulers*. Then we say that a probabilistic security property holds for a given model, if the property holds after resolving the nondeterminism under *all admissible schedulers*.

2 Probabilistic Automata

In this section we gather preliminary notions and results related to probabilistic automata [SL94, Seg95]. Some of the formulations we borrow from [Sok05] and [Che06]. We shall model protocols with probabilistic automata. We start with a definition of probability distribution.

Definition 2.1 (Probability distribution). A function $\mu: S \rightarrow [0, 1]$ is a discrete probability distribution, or distribution for short, on a set S if $\sum_{x \in S} \mu(x) = 1$. The set $\{x \in S \mid \mu(x) > 0\}$ is the support of μ and is denoted by $\text{supp}(\mu)$. By $\mathcal{D}(S)$ we denote the set of all discrete probability distributions on the set S .

We use the simple probabilistic automata [SL94, Seg95], or MDP’s [Bel57] as models of our probabilistic processes. These models are very similar to the labelled transition systems, with the only difference that the target of each transition is a distribution over states instead of just a single next state.

Definition 2.2 (Probabilistic automaton). A probabilistic automaton is a triple $\mathcal{A} = \langle S, A, \alpha \rangle$ where:

- S is a set of states.
- A is a set of actions or action labels.
- α is a transition function $\alpha: S \rightarrow \mathcal{P}(A \times \mathcal{D}S)$.

A terminating state of \mathcal{A} is a state with no outgoing transition, i.e. with $\alpha(s) = \emptyset$. We might sometimes also specify an initial state $s_0 \in S$ of a probabilistic automaton \mathcal{A} . We write $s \xrightarrow{a} \mu$ for $(a, \mu) \in \alpha(s)$, $s \in S$. Moreover, we write $s \xrightarrow{a, \mu} t$ for $s, t \in S$ whenever $s \xrightarrow{a} \mu$ and $\mu(t) > 0$.

We will also need the notion of a fully probabilistic system.

Definition 2.3 (Fully Probabilistic Automaton). A fully probabilistic automaton is a triple $\mathcal{A} = \langle S, A, \alpha \rangle$ where:

- S is a set of states.
- A is a set of actions or action labels.
- α is a transition function $\alpha : S \rightarrow \mathcal{D}(A \times S) + 1$.

Here $1 = \{*\}$ denotes termination, i.e., if $\alpha(s) = *$ then s is a terminating state. It can also be understood as a zero-distribution i.e. $\alpha(s)(a, t) = 0$ for all $a \in A$ and $t \in S$. By $s_0 \in S$ we sometimes denote an initial state of \mathcal{A} . We write $s \rightarrow \mu$ for $\mu = \alpha(s)$, $s \in S$. Moreover, we write $s \xrightarrow{a} t$ for $s, t \in S$ whenever $s \rightarrow \mu$ and $\mu(a, t) > 0$.

A major difference between the (simple) probabilistic automata and the fully probabilistic ones is that the former can express nondeterminism. In order to reason about probabilistic properties of a model with nondeterminism we first resolve the nondeterminism with help of schedulers or adversaries – this leaves us with a fully probabilistic model whose probabilistic behaviour we can analyze. We explain this in the sequel.

Definition 2.4 (Paths). A path of a *probabilistic automaton* \mathcal{A} is a sequence

$$\pi = s_0 \xrightarrow{a_1, \mu_1} s_1 \xrightarrow{a_2, \mu_2} s_2 \dots$$

where $s_i \in S$, $a_i \in A$ and $s_i \xrightarrow{a_{i+1}, \mu_{i+1}} s_{i+1}$.

A path of a *fully probabilistic automaton* \mathcal{A} is a sequence

$$\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \dots$$

where again $s_i \in S$, $a_i \in A$ and $s_i \xrightarrow{a_{i+1}} s_{i+1}$.

A path can be finite in which case it ends with a state. A path is complete if it is either infinite or finite ending in a terminating state. We let $\text{last}(\pi)$ denote the last state of a finite path π , and for arbitrary path $\text{first}(\pi)$ denotes its first state. A trace of a path is the sequence of actions in $A^* \cup A^\infty$ obtained by removing the states (and the distributions), hence above $\text{trace}(\pi) = a_1 a_2 \dots$. The length of a finite path π , denoted by $|\pi|$ is the number of actions in its trace. Let $\text{Paths}(\mathcal{A})$ denote the set of all paths, $\text{Paths}^{\leq \omega}(\mathcal{A})$ the set of all finite paths, and $\text{CPaths}(\mathcal{A})$ the set of all complete paths of an automaton \mathcal{A} .

Paths are ordered by the prefix relation, which we denote by \leq .

Let \mathcal{A} be a (fully) probabilistic automaton and let π_i for $i \geq 0$ be finite paths of \mathcal{A} all starting in the same initial state s_0 and such that $\pi_i \leq \pi_j$ for $i \leq j$ and $|\pi_i| = i$, for all $i \geq 0$. Then by $\pi = \lim_{i \rightarrow \infty} \pi_i$ we denote the infinite complete path with the property that $\pi_i \leq \pi$ for all $i \geq 0$.

Definition 2.5 (Cone). Let \mathcal{A} be a (fully) probabilistic automaton and let $\pi \in \text{Paths}^{\leq \omega}(\mathcal{A})$ be given. The cone generated by π is the set of paths

$$C_\pi = \{\pi' \in \text{CPaths}(\mathcal{A}) \mid \pi \leq \pi'\}.$$

From now on we fix an initial state. Given a fully probabilistic automaton \mathcal{A} with an initial state s_0 , we can calculate the probability-value denoted by $\mathbf{P}(\pi)$

of any finite path π starting in s_0 as follows.

$$\begin{aligned}\mathbf{P}(s_0) &= 1 \\ \mathbf{P}(\pi \xrightarrow{a} s) &= \mathbf{P}(\pi) \cdot \mu(a, s) \quad \text{where } \text{last}(\pi) \rightarrow \mu\end{aligned}$$

Let $\Omega_{\mathcal{A}} = \text{CPaths}(\mathcal{A})$ be the sample space, and let $\mathcal{F}_{\mathcal{A}}$ be the smallest σ -algebra generated by the cones. The following proposition (see [Seg95, Sok05]) states that \mathbf{P} induces a unique probability measure on $\mathcal{F}_{\mathcal{A}}$.

Proposition 1. *Let \mathcal{A} be a fully probabilistic automaton and let \mathbf{P} denote the probability-value on paths. There exists a unique probability measure on $\mathcal{F}_{\mathcal{A}}$ also denoted by \mathbf{P} such that $\mathbf{P}(C_{\pi}) = \mathbf{P}(\pi)$ for every finite path π . \square*

This way we are able to measure the probability of certain events described by sets of paths in an automaton with no nondeterminism. Since our models include nondeterminism, we will first resolve it by means of schedulers or adversaries. Before we define adversaries note that we can describe the set of all sub-probability distributions on a set S by $\mathcal{D}(S + 1)$. These are functions whose sum of values on S is not necessarily equal to 1, but it is bounded by 1.

Definition 2.6 (Scheduler). A scheduler for a probabilistic automaton \mathcal{A} is a function

$$\xi: \text{Paths}^{\leq \omega}(\mathcal{A}) \rightarrow \mathcal{D}(A \times \mathcal{D}(S) + 1)$$

satisfying $\xi(\pi)(a, \mu) > 0$ implies $\text{last}(\pi) \xrightarrow{a} \mu$, for each finite path π . By $\text{Sched}(\mathcal{A})$ we denote the set of all schedulers for \mathcal{A} .

Hence, a scheduler according to the previous definition imposes a probability distribution on the possible non-deterministic transitions in each state. Therefore it is randomized. It is history dependent since it takes into account the path (history) and not only the current state. It is partial since it gives a sub-probability distribution, i.e., it may halt the execution at any time.

Definition 2.7 (Automaton under scheduler). A probabilistic automaton $\mathcal{A} = \langle S, A, \alpha \rangle$ together with a scheduler ξ determine a fully probabilistic automaton

$$\mathcal{A}_{\xi} = \langle \text{Paths}^{\leq \omega}(\mathcal{A}), A, \alpha_{\xi} \rangle.$$

Its set of states are the finite paths of \mathcal{A} , its initial state is the initial state of \mathcal{A} (seen as a path with length 1), its actions are the same as those of \mathcal{A} , and its transition function α_{ξ} is defined as follows. For any $\pi \in \text{Paths}^{\leq \omega}(\mathcal{A})$, we have $\alpha_{\xi}(\pi) \in \mathcal{D}(A \times \text{Paths}^{\leq \omega}(\mathcal{A})) + 1$ as

$$\alpha_{\xi}(\pi)(a, \pi') = \begin{cases} \xi(\pi)(a, \mu) \cdot \mu(s) & \pi' = \pi \xrightarrow{a, \mu} s \\ 0 & \text{otherwise} \end{cases}$$

Given a probabilistic automaton \mathcal{A} and a scheduler ξ , we denote by \mathbf{P}_{ξ} the probability measure on sets of complete paths of the fully probabilistic automaton \mathcal{A}_{ξ} , as in Proposition 1. The corresponding σ -algebra generated by cones of finite paths of \mathcal{A}_{ξ} we denote by Ω_{ξ} . The elements of Ω_{ξ} are measurable sets.

By Ω we denote the σ -algebra generated by cones of finite paths of \mathcal{A} (without fixing the scheduler!) and also call its elements measurable sets, without having a measure in mind. Actually, we will now show that any scheduler $\xi \in \text{Sched}(\mathcal{A})$ induces a measure $\mathbf{P}_{(\xi)}$ on a certain σ -algebra $\Omega_{(\xi)}$ of paths in \mathcal{A} such that $\Omega \subseteq \Omega_{(\xi)}$. Hence, any element of Ω can be measured by any of these measures $\mathbf{P}_{(\xi)}$. We proceed with the details.

Define a function $f : \text{Paths}^{\leq \omega}(\mathcal{A}_\xi) \rightarrow \text{Paths}^{\leq \omega}(\mathcal{A})$ by

$$f(\hat{\pi}) = \text{last}(\hat{\pi}) \quad (1)$$

for any $\hat{\pi} \in \mathcal{A}_\xi$. The function f is well-defined since states in \mathcal{A}_ξ are the finite paths of \mathcal{A} . Moreover, we have the following property.

Lemma 1. *For any $\hat{\pi}_1, \hat{\pi}_2 \in \text{Paths}^{\leq \omega}(\mathcal{A}_\xi)$ we have*

$$\hat{\pi}_1 \leq \hat{\pi}_2 \iff f(\hat{\pi}_1) \leq f(\hat{\pi}_2)$$

where the order on the left is the prefix order in $\text{Paths}^{\leq \omega}(\mathcal{A}_\xi)$ and on the right the prefix order in $\text{Paths}^{\leq \omega}(\mathcal{A})$.

Proof. By the definition of \mathcal{A}_ξ we have that for $\pi, \pi' \in \text{Paths}^{\leq \omega}(\mathcal{A})$ i.e. states of \mathcal{A}_ξ : $\pi \xrightarrow{a} \pi'$ if and only if $\xi(\pi) > 0$ and $\text{last}(\pi) \xrightarrow{a, \mu} s$ in \mathcal{A} for some μ and s . In other words if $\pi \xrightarrow{a} \pi'$, then $\pi \leq \pi'$ and $|\pi'| = |\pi| + 1$ i.e. π' extends π in one step. Therefore, if we have a path $\pi_0 \xrightarrow{a_0} \pi_1 \xrightarrow{a_1} \pi_2 \xrightarrow{a_2} \dots$ in \mathcal{A}_ξ , then for all its states: if $i \leq j$, then $\pi_i \leq \pi_j$ and $|\pi_j| = |\pi_i| + (j - i)$. So if $\hat{\pi}_1, \hat{\pi}_2 \in \text{Paths}^{\leq \omega}(\mathcal{A}_\xi)$ are such that $\hat{\pi}_1 \leq \hat{\pi}_2$, then $\text{last}(\hat{\pi}_1)$ is a state in $\hat{\pi}_2$ and therefore we at once get $\text{last}(\hat{\pi}_1) \leq \text{last}(\hat{\pi}_2)$. For the opposite implication, again from the definition we notice that if a path $\hat{\pi} \in \text{Paths}^{\leq \omega}(\mathcal{A})$ contains a state $\pi \in \text{Paths}^{\leq \omega}(\mathcal{A})$, then it also contains all prefixes of π as states. Hence, if $\text{last}(\hat{\pi}_1) \leq \text{last}(\hat{\pi}_2)$ for $\hat{\pi}_1, \hat{\pi}_2 \in \text{Paths}^{\leq \omega}(\mathcal{A}_\xi)$, then $\text{last}(\hat{\pi}_1)$ is a state in $\hat{\pi}_2$ and also all its prefixes are. Since all paths start in the initial state (path), this implies that $\hat{\pi}_1 \leq \hat{\pi}_2$. \square

Corollary 1. *The function f defined by (1) is injective.* \square

By Lemma 1 we can extend the function f to $\hat{f} : \text{CPaths}(\mathcal{A}_\xi) \rightarrow \text{CPaths}(\mathcal{A})$ by

$$\hat{f}(\hat{\pi}) = \begin{cases} f(\hat{\pi}) & \pi \text{ is finite} \\ \lim_{i \rightarrow \infty} f(\hat{\pi}_i) & \hat{\pi}_i \leq \pi, |\hat{\pi}_i| = i \end{cases}$$

The properties from Lemma 1 and Corollary 1 continue to hold for the extended function \hat{f} as well. We will write f for \hat{f} as well.

Recall that Ω_ξ denotes the σ -algebra on which the measure \mathbf{P}_ξ is defined. We now define a family of subsets Ω^ξ of $\text{CPaths}(\mathcal{A})$ by

$$\Omega^\xi = \{\Pi \in \text{CPaths}(\mathcal{A}) \mid f^{-1}(\Pi) \in \Omega_\xi\}. \quad (2)$$

The following properties are instances of standard measure-theoretic results.

Lemma 2. *The family Ω^ξ is a σ -algebra on $\text{CPaths}(\mathcal{A})$ and by*

$$\mathbf{P}^\xi(\Pi) = \mathbf{P}_\xi(f^{-1}(\Pi))$$

for $\Pi \in \Omega^\xi$ a measure on Ω^ξ is defined.

Recall that Ω denotes the σ -algebra on complete paths of \mathcal{A} generated by the cones. We show that for any scheduler ξ , $\Omega \subseteq \Omega^\xi$. Hence, the measurable sets (elements of Ω) are indeed measurable by the measure induced by any scheduler.

Lemma 3. *For any scheduler ξ , $\Omega \subseteq \Omega^\xi$.*

Proof. Fix a scheduler ξ . Since Ω is generated by the cones it is enough to show that each cone is in Ω^ξ . Let $C_{\pi_0, \mathcal{A}}$ be a cone in $\text{CPaths}(\mathcal{A})$ generated by the finite path π_0 , i.e.

$$C_{\pi_0, \mathcal{A}} = \{\pi \in \text{CPaths}(\mathcal{A}) \mid \pi_0 \leq \pi\}.$$

We have

$$\hat{f}^{-1}(C_{\pi_0, \mathcal{A}}) = \begin{cases} \emptyset & \pi_0 \notin \hat{f}(\text{CPaths}(\mathcal{A})) \\ C_{\hat{\pi}_0, \mathcal{A}_\xi} & \hat{f}(\hat{\pi}_0) = \pi_0 \end{cases}.$$

by Lemma 1. Indeed, let $\pi_0 = \hat{f}(\hat{\pi}_0)$. Then

$$\begin{aligned} \hat{f}^{-1}(C_{\pi_0, \mathcal{A}}) &= \{\hat{\pi} \in \text{CPaths}(\mathcal{A}_\xi) \mid \hat{f}(\hat{\pi}) \geq \pi_0\} \\ &= \{\hat{\pi} \in \text{CPaths}(\mathcal{A}_\xi) \mid \hat{f}(\hat{\pi}) \geq \hat{f}(\hat{\pi}_0)\} \\ (\text{Lem. 1}) &= \{\hat{\pi} \in \text{CPaths}(\mathcal{A}_\xi) \mid \hat{\pi} \geq \hat{\pi}_0\} \\ &= C_{\hat{\pi}_0, \mathcal{A}_\xi} \end{aligned}$$

We next define two operations on probabilistic automata used for building composed models out of basic models: parallel composition and restriction. We compose probabilistic automata in parallel in the style of the process algebra ACP. That is, asynchronously with communication function given by a semigroup operation on the set of actions. This is the most general way of composing probabilistic automata in parallel (for an overview see [SV04]).

Definition 2.8 (Parallel composition). We fix an action set A and a communication function \cdot on A which is a partial commutative semigroup. Given two probabilistic automata $\mathcal{A}_1 = \langle S_1, A, \alpha_1 \rangle$ and $\mathcal{A}_2 = \langle S_2, A, \alpha_2 \rangle$ with actions A , their parallel composition is the probabilistic automaton $\mathcal{A}_1 \parallel \mathcal{A}_2 = \langle S_1 \times S_2, A, \alpha \rangle$ with states pairs of states of the original automata denoted by $s_1 \parallel s_2$, the same actions, and transition function defined as follows. $s_1 \parallel s_2 \xrightarrow{a} \mu$ if and only if one of the following holds

1. $s_1 \xrightarrow{b} \mu_1$ and $s_2 \xrightarrow{c} \mu_2$ for some actions b and c such that $a = b \cdot c$ and $\mu = \mu_1 \cdot \mu_2$ meaning $\mu(t_1 \parallel t_2) = \mu_1(t_1) \cdot \mu_2(t_2)$.
2. $s_1 \xrightarrow{a} \mu'$ where $\mu'(t_1) = \mu(t_1 \parallel s_2)$ for all states t_1 of the first automaton.
3. $s_2 \xrightarrow{a} \mu'$ where $\mu'(t_2) = \mu(s_1 \parallel t_2)$ for all states t_2 of the second automaton.

Here, 1. represents a synchronous joint move of both of the automata, and 2. and 3. represent the possibilities of an asynchronous move of each of the automata. In case s_1^0 and s_2^0 are the initial states of \mathcal{A}_1 and \mathcal{A}_2 , respectively, then the initial state of $\mathcal{A}_1 \parallel \mathcal{A}_2$ is $s_1^0 \parallel s_2^0$.

Often we will use input and output actions like $a?$ and $a!$, respectively, in the style of CCS. In such cases we assume that the communication is defined as handshaking $a? \cdot a! = \tau_a$ for τ_a a special invisible action.

The operation of restriction is needed to prune out some branches of a probabilistic automaton that one need not consider. For example, we will commonly use restriction to get rid of parts of a probabilistic automaton that still wait on synchronization.

Definition 2.9 (restriction). Fix a subset $I \subseteq A$ of actions that are in the restricted set. Given an automaton $\mathcal{A} = \langle S, A, \alpha \rangle$, the automaton obtained from \mathcal{A} by restricting the actions in I is $\mathcal{R}_I(\mathcal{A}) = \langle S, A \setminus I, \alpha' \rangle$ where the transitions of α' are defined as follows: $s \xrightarrow{a} \mu$ in $\mathcal{R}_I(\mathcal{A})$ if and only if $s \xrightarrow{a} \mu$ in \mathcal{A} and $a \notin I$.

We now define bisimilarity - a behaviour equivalence on the states of a probabilistic automaton. For that we first need the notion of relation lifting.

Definition 2.10 (Relation lifting). Let R be an equivalence relation on the set of states S of a probabilistic automaton \mathcal{A} . Then R lifts to a relation \equiv_R on the set $\mathcal{D}(S)$, as follows:

$$\mu \equiv_R \nu \iff \sum_{s \in C} \mu(s) = \sum_{s \in C} \nu(s)$$

for any equivalence class $C \in S/R$.

Definition 2.11 (Bisimulation, bisimilarity). Let $\mathcal{A} = \langle S, A, \alpha \rangle$ be a probabilistic automaton. An equivalence R on its set of states S is a bisimulation if and only if whenever $\langle s, t \rangle \in R$ we have

$$\text{if } s \xrightarrow{a} \mu_s, \text{ then there exists } \mu_t \text{ such that } t \xrightarrow{a} \mu_t \text{ and } \mu_s \equiv_R \mu_t.$$

Two states $s, t \in S$ are bisimilar, notation $s \sim t$ if they are related by some bisimulation relation R .

Note that bisimilarity \sim is the largest bisimulation on a given probabilistic automaton \mathcal{A} .

3 Anonymizing Protocols

3.1 Dining cryptographers

The canonical example of an anonymizing protocol is Chaum's Dining Cryptographers [Cha88]. In Chaum's introduction to this protocol, three cryptographers are sitting down to dine in a restaurant, when the waiter informs them that the bill has already been paid anonymously. They wonder whether one of them has paid the bill in advance, or whether the NSA has done so. Respecting each other's

right to privacy, they carry out the following protocol. Each pair of cryptographers flips a coin, invisible to the remaining cryptographer. Each cryptographer then reveals whether or not the two coins he saw were equal or unequal. However, if a cryptographer is paying, he states the opposite. An even number of “equals” now indicates that the NSA is paying; an odd number that one of the cryptographers is paying.

Formally, Chaum states the result as follows. (Here we are restricting to the case with 3 cryptographers; Chaum’s version is more general.) Here, \mathbb{F}_2 is the field of two elements.

Theorem 3.1 *Let K be a uniformly distributed stochastic variable over \mathbb{F}_2^3 . Let I be a stochastic variable over \mathbb{F}_2^3 , taking only values in $\{(1, 0, 0), (0, 1, 0), (0, 0, 1), (0, 0, 0)\}$. Let A be the stochastic variable over \mathbb{F}_2^3 given by $A = (I_1 + K_2 + K_3, K_1 + I_2 + K_3, K_1 + K_2 + I_3)$. Assume that K and I are independent. Then*

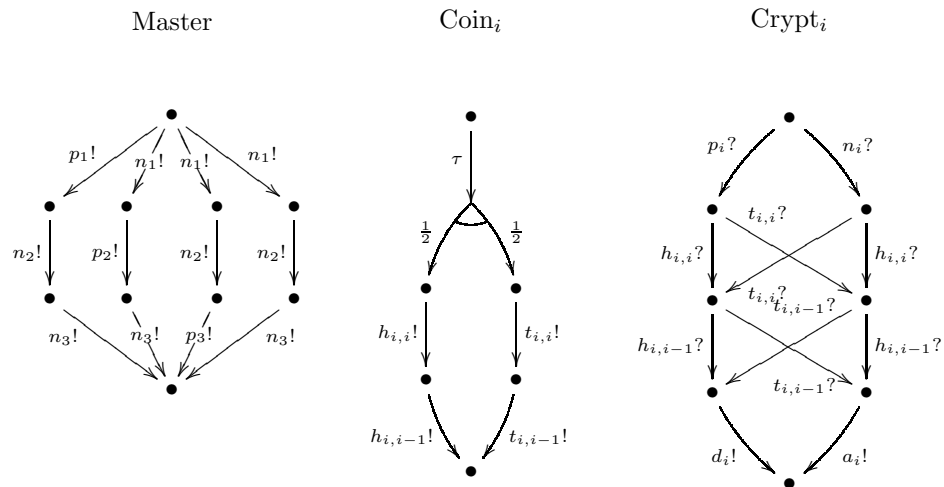
$$\forall a \in \mathbb{F}_2^3 \forall i \in \{1, 2, 3\} : \mathbb{P}[I = i] > 0 \implies \mathbb{P}[A = a \mid I = i] = \frac{1}{4}$$

and hence

$$\forall a \in \mathbb{F}_2^3 \forall i \in \{1, 2, 3\} : \mathbb{P}[I = i] > 0 \implies \mathbb{P}[A = a \mid I = i] = \mathbb{P}[A = a]. \quad \square$$

In terms of the storyline, K represents the coin flips, I represents which cryptographer (if any) is paying, and A represents the every cryptographer says.

We will now model this protocol as a probabilistic automaton. We will construct it as a parallel composition of seven components: the Master, who decides who will pay, the three cryptographers Crypt_i , and the three coins Coin_i . The action $p_i!$ is used by the Master to indicate to Crypt_i that he should pay; the action $n_i!$ to indicate that he shouldn’t. If no cryptographer is paying, the NSA is paying, which is not explicitly modelled here. The coin Coin_i is shared by Crypt_i and Crypt_{i-1} (taking the -1 modulo 3); the action $h_{i,j}!$ represents Coin_i signalling to Crypt_j that the coin was heads and similarly $t_{i,j}!$ signals tails. At the end, the cryptographers state whether or not the two coins they saw were equal or not by means of the actions $a_i!$ (agree) or $d_i!$ (disagree).



Now DC is the parallel composition of Master, Coin₀, Coin₁, Coin₂, Crypt₀, Crypt₁, and Crypt₂ with all actions of the form p_i , n_i , $h_{i,j}$, and $t_{i,j}$ hidden.

Note that in Chaum’s version, there is no assumption on the probability distribution of I ; in our version this is modelled by the fact that the Master makes a non-deterministic choice between the four options. Since we allow probabilistic schedulers, we later recover all possible probability distributions about who is paying, just as in the original version. Independence between the choice of the master and the coin flips (I and K in Chaum’s version) comes for free in the automata model: distinct probabilistic choices are always assumed to be independent.

In Section 4 we formulate what it means for DC (or more general, for an anonymity automaton) to be anonymous.

3.2 Voting

At a very high level, a voting protocol can be seen as a blackbox that inputs the voters’ votes and outputs the result of the vote. For simplicity, assume the voters vote yes (1) or no (0), do not abstain, and that the numbers of voters is known. The result then is the number of yes-votes.



In such a setting, it is conceivable that an observer has some a-priori knowledge about which voters are more likely to vote yes and which voters are more likely to vote no. Furthermore, there definitely is a-posteriori knowledge, since the vote result is made public. For instance, in the degenerate case where all voters vote the same way, everybody’s vote is revealed. What we expect here from the voting protocol is not that the adversary has no knowledge about the votes (since he might already have a-priori knowledge), and also not that the adversary does not gain any knowledge from observing the protocol (since the vote result is revealed), but rather that observing the protocol does not augment the adversary’s knowledge beyond learning the vote result.

For the purely probabilistic case, this notion of anonymity is formalized in Section 4.

4 Anonymity for Purely Probabilistic Systems

This section defines anonymity systems and proposes a definition for anonymity in its simplest configuration, i.e., for purely probabilistic systems. Purely probabilistic systems are simpler because there is no need for schedulers. Throughout the following sections, this definitions will be incrementally modified towards a more general setting.

Definition 4.1 (Anonymity system). Let $M = \langle S, \text{Act}, \alpha \rangle$ be a fully probabilistic automaton. An *anonymity system* is a triple $\langle M, I, \{A_i\}_{i \in I}, \text{Act}_O \rangle$ where

1. I is the set of user identities,
2. A_i is any measurable subset of $\text{CPaths}(M)$ such that $A_i \cap A_j = \emptyset$ for $i \neq j$.
3. $\text{Act}_O \subseteq \text{Act}$ is the set of observable actions.
4. $\text{Otrace}(\pi)$ is the sequence of elements in Act_O obtained by removing from $\text{trace}(\pi)$ the elements in $\text{Act} \setminus \text{Act}_O$.

Define O as the set of observations, i.e., $O = \{\text{trace}(\pi) \mid \pi \in \text{Paths}(M)\}$. We also define $A = \bigcup_{i \in I} A_i$.

Intuitively, the A_i s are properties of the executions that the system is meant to hide. For example, in the case of the dining cryptographers A_i would be “cryptographer i payed”; in a voting scheme “voter i voted for candidate c ”, etc. Therefore, for the previous examples, the predicate A would be “some of the cryptographers payed” or “the vote count” respectively.

Next, we propose a definition of anonymity for a purely probabilistic systems. We deviate from the definition proposed by Bhargava and Palamidessi [BP05] for what we consider a more intuitive definition: We say that an anonymity system is anonymous if the probability of seeing a observation is independent of who performed the anonymous action (A_i), given that some anonymous action took place (A happened). The formal definition follows.

Definition 4.2 (Anonymity). A system $\langle M, I, \{A_i\}_{i \in I}, \text{Act}_O \rangle$ is said to be anonymous if

$$\forall i \in I. \forall o \in O. \mathbb{P}[\pi \in A] > 0 \implies \mathbb{P}[\text{Otrace}(\pi) = o \wedge \pi \in A_i \mid \pi \in A] = \mathbb{P}[\text{Otrace}(\pi) = o \mid \pi \in A] \mathbb{P}[\pi \in A_i \mid \pi \in A].$$

In the above probabilities, π is drawn from the probability space $\text{Paths}(M)$.

The following lemma shows that this definition is equivalent to the one proposed in Bhargava and Palamidessi [BP05].

Lemma 4. *A anonymity system is anonymous if and only if*

$$\forall i, j \in I. \forall o \in O. (\mathbb{P}[\pi \in A_i] > 0 \wedge \mathbb{P}[\pi \in A_j] > 0) \implies \mathbb{P}[\text{Otrace}(\pi) = o \mid \pi \in A_i] = \mathbb{P}[\text{Otrace}(\pi) = o \mid \pi \in A_j]$$

Proof. The only if part is trivial. For the if part we have

$$\begin{aligned}
& \mathbb{P}[Otrace(\pi) = o \mid \pi \in A] \mathbb{P}[\pi \in A_i \mid \pi \in A] \\
&= \mathbb{P}[\pi \in A_i \mid \pi \in A] \sum_{j \in I} \mathbb{P}[Otrace(\pi) = o \mid \pi \in A_j \cap A] \mathbb{P}[\pi \in A_j \mid \pi \in A] \\
&\quad (\text{since } A_i \cap A_j = \emptyset, i \neq j) \\
&= \mathbb{P}[\pi \in A_i \mid \pi \in A] \sum_{j \in I} \mathbb{P}[Otrace(\pi) = o \mid \pi \in A_j] \mathbb{P}[\pi \in A_j \mid \pi \in A] \\
&\quad (\text{by definition of } \pi \in A) \\
&= \mathbb{P}[\pi \in A_i \mid \pi \in A] \mathbb{P}[Otrace(\pi) = o \mid \pi \in A_i] \sum_{j \in I} \mathbb{P}[\pi \in A_j \mid \pi \in A] \\
&\quad (\text{by hypothesis}) \\
&= \mathbb{P}[\pi \in A_i \mid \pi \in A] \mathbb{P}[Otrace(\pi) = o \mid \pi \in A_i] \frac{\sum_{j \in I} \mathbb{P}[\pi \in A_j]}{\mathbb{P}[\pi \in A]} \\
&\quad (\text{since } A_j \subseteq A) \\
&= \mathbb{P}[\pi \in A_i \mid \pi \in A] \mathbb{P}[Otrace(\pi) = o \mid \pi \in A_i] \\
&= \frac{\mathbb{P}[\pi \in A_i]}{\mathbb{P}[\pi \in A]} \frac{\mathbb{P}[Otrace(\pi) = o \wedge \pi \in A_i]}{\mathbb{P}[\pi \in A_i]} \\
&= \mathbb{P}[Otrace(\pi) = o \wedge \pi \in A_i \mid \pi \in A] \\
&\quad (\text{since } A_i \subseteq A)
\end{aligned}$$

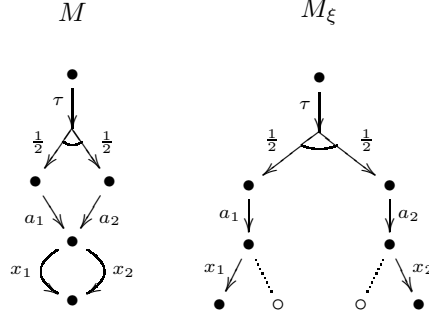
which concludes the proof. \square

5 Anonymity for Probabilistic Systems

We now try to extend the notion of anonymity to probabilistic automata that are not purely probabilistic, but that still contain some non-deterministic transitions.

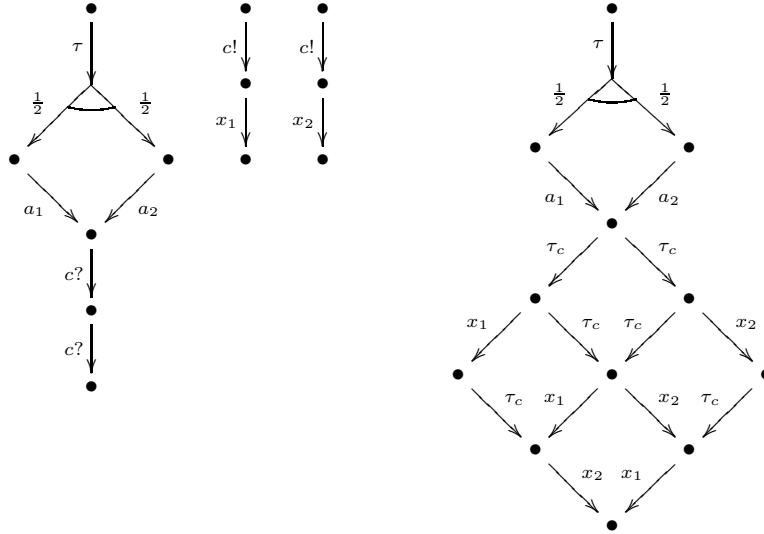
One obvious try is to say that M is anonymous if M_ξ is anonymous for all schedulers ξ of M . The following automaton M and scheduler ξ show that this

definition would be problematic.



Here a_1 and a_2 are invisible actions; they represent which user performed the action that was to remain hidden. The actions x_1 and x_2 are observable. Intuitively, because the adversary cannot see the messages a_1 and a_2 , she cannot learn which user actually performed the hidden action. On the right hand side M_ξ is shown and the branches the scheduler does not take are indicated by dotted arrows. Now $\mathbb{P}_\xi[a_1 \mid x_1] = 1$, but $\mathbb{P}_\xi[a_1] = \frac{1}{2}$, showing that with this particular scheduler M_ξ is not anonymous.

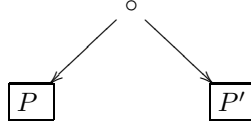
Note that this phenomenon can easily occur as a consequence of communication non-determinism. For instance, consider the following three automata and their parallel composition in which $c?$ and $c!$ are hidden. In this example the order of the messages x_1 and x_2 depends on a race-condition, but a scheduler can make it depend on whether a_1 or a_2 was taken. I.e., there exists a scheduler ξ such that $\mathbb{P}_\xi[x_1x_2 \mid a_1] = \mathbb{P}_\xi[x_2x_1 \mid a_2] = 1$ and hence $\mathbb{P}_\xi[x_2x_1 \mid a_1] = \mathbb{P}_\xi[x_1x_2 \mid a_2] = 0$.



In fact, the Dining Cryptographers example from Section 3.1 suffers from exactly the same problem. The order in which the cryptographers say *agree_i* or *disagree_i*

is determined by the scheduler and it is possible to have a scheduler that makes the paying cryptographer, if any, go last.

In [BP05], a system M is called anonymous if for all schedulers ζ, ξ , for all observables o , and for all hidden actions a_i, a_j such that $\mathbb{P}_\zeta[a_i] > 0$ and $\mathbb{P}_\xi[a_j] > 0$, $\mathbb{P}_\zeta[o \mid a_i] = \mathbb{P}_\xi[o \mid a_j]$. This definition, of course, has the same problems as above; in the Dining Cryptographers example in [BP05] this is solved by fixing the order in which the cryptographers say *agree_i* or *disagree_i*. However, also a non-deterministic choice between two otherwise anonymous systems can become non-anonymous with this definition. For instance, let P be some anonymous system. For simplicity, assume that P is fully probabilistic (e.g., the Dining Cryptographers with a probabilistic master and a fixed scheduler) and let P' be a variant of P in which the visible actions have been renamed (e.g., the actions *agree_i* and *disagree_i* are renamed to *equal_i* and *unequal_i*). Now consider the probabilistic automaton M which non-deterministically chooses between P and P' :



This automaton has only two schedulers: the one that chooses the left branch and then executes P and the one that chooses the right branch and then executes P' . Let's call these schedulers l and r respectively. Now pick any hidden action a_i and observable o such that $\mathbb{P}_l[o \mid a_i] > 0$. (e.g., $o = \text{agree}_1 \text{disagree}_2 \text{agree}_3$ and $a_i = \text{pay}_1$, for which $\mathbb{P}[o \mid \text{pay}_1] = \frac{1}{4}$). Then, nevertheless, $\mathbb{P}_r[o \mid a_i] = 0$, because the observation o cannot occur in P' . So, even though intuitively this system should be anonymous, it is not so according to the definition in [BP05].

Every time the problem is that the scheduler has access to information it shouldn't have. When one specifies a protocol by giving a probabilistic automaton, an implementation of this protocol has to implement a scheduler as well. This is especially obvious if the non-determinism originates from communication. When we identify schedulers with adversaries, as is common, it becomes clear that the scheduler should not have access to too much information. In the next section we define a class of schedulers, called *admissible* schedulers that base their scheduling behavior on the information an adversary actually has access to: the observable history of the system.

6 Admissible Schedulers

As explained in the previous section, defining anonymity as a condition that should hold true for all possible schedulers is problematic. It is usual to quantify over all schedulers when showing theoretical properties of systems with both probabilities and non-determinism - for example we may say "no matter how the non-determinism is resolved, the probability of an event X is at least p ". However, in the analysis of security protocols, for example with respect to anonymity,

we would like to quantify over all possible “realistic” adversaries. These are not all possible schedulers as in our theoretic considerations since such a realistic adversary is not able to see all details of the probabilistic automaton under consideration. Hence, considering that the adversary is any scheduler enables the adversary to leak information where it normally could not. We call such schedulers interfering schedulers. This way protocols that are well-known to be anonymous turn out not to be anonymous. One such example is the dining cryptographers protocol explained above. We show that one gets a better definition of anonymity if one restricts the power of the schedulers, in a realistic way. In this section we define the type of schedulers with restricted power that we consider good enough for showing anonymity of certain protocols. We call these schedulers admissible.

Schedulers with restricted power have been treated in the literature. In general, as explained by Segala in [Seg95], a scheduler with restricted power is given by defining two equivalences, one on the set of finite paths \equiv_1 and another one \equiv_2 on the set of possible transition, in this case $\mathcal{D}(A \times S)$. Then a scheduler ξ is oblivious relative to $\langle \equiv_1, \equiv_2 \rangle$ if and only if for any two paths π_1, π_2 we have

$$\pi_1 \equiv_1 \pi_2 \implies \xi(\pi_1) \equiv_2 \xi(\pi_2).$$

6.1 Admissible schedulers based on bisimulation

In this section we specify \equiv_1 and \equiv_2 and obtain a class of oblivious adversaries that suits the anonymity definition.

Defined \equiv_1 on the set of finite paths of an automaton M as,

$$\pi_1 \equiv_1 \pi_2 \iff (\text{trace}(\pi_1) = \text{trace}(\pi_2) \wedge \text{last}(\pi_1) \sim \text{last}(\pi_2)).$$

Recall that we defined \equiv_R as the lifting of the equivalence relation R on a set S to an equivalence relation on $\mathcal{D}(A \times S)$. For \equiv_2 we take the equivalence \equiv_\sim on $\mathcal{D}(A \times S)$. This is well defined since bisimilarity is an equivalence. Hence, we obtain a class of oblivious schedulers relative $\langle \equiv_1, \equiv_\sim \rangle$. These schedulers we call admissible.

Definition 6.1 (admissible scheduler). A scheduler is admissible if for any two finite paths π_1 and π_2 we have

$$(\text{trace}(\pi_1) = \text{trace}(\pi_2) \wedge \text{last}(\pi_1) \sim \text{last}(\pi_2)) \implies \xi(\pi_1) \equiv_\sim \xi(\pi_2).$$

Intuitively, the definition of a admissible scheduler enforces that in cases when the scheduler has observed the same history (given by the traces of the paths) and is in bisimilar states, it must schedule “the same” transitions up to bisimilarity.

6.2 Existence

We now show that admissible schedulers do exist. In fact, we even show that admissible history-independent schedulers exist. A scheduler ξ is history-independent if it is completely determined by its image of paths of length 0 i.e. if for any path π it holds that $\xi(\pi) = \xi(\text{last}(\pi))$.

Theorem 6.2 (Existence) *There exists a admissible scheduler for every probabilistic automaton.*

Proof. Take a probabilistic automaton M . We first show that there exists a map $\xi : S \rightarrow \mathcal{D}(A \times S) \cup \{\perp\}$ with the property that $\xi(s) = \perp$ if and only if s terminates and for all $s, t \in S$, if $s \sim t$, then $\xi(s) \equiv_{\sim} \xi(t)$.

Consider the set of partial maps

$$\Xi = \left\{ \xi : S \hookrightarrow \mathcal{D}(A \times S) \cup \{\perp\} \left| \begin{array}{l} \xi(s) = \perp \iff s \text{ terminates} , \\ s \sim t \implies \xi(s) \equiv_{\sim} \xi(t) \\ \text{for } s, t \in \text{dom}(\xi) \end{array} \right. \right\}.$$

This set is not empty since the unique partial map with empty domain belongs to it. We define an order \leq on Ξ in a standard way by

$$\xi_1 \leq \xi_2 \iff (\text{dom}(\xi_1) \subseteq \text{dom}(\xi_2) \wedge \xi_2|_{\text{dom}(\xi_1)} = \xi_1).$$

Consider a chain $(\xi_i)_{i \in I}$ in Ξ . Let $\xi = \cup_{i \in I} \xi_i$. This means that $\text{dom}(\xi) = \cup_{i \in I} \text{dom}(\xi_i)$ and if $x \in \text{dom}(\xi)$, then $\xi(x) = \xi_i(x)$ for $i \in I$ such that $x \in \text{dom}(\xi_i)$. Note that ξ is well-defined since $(\xi_i)_{i \in I}$ is a chain. Moreover, it is obvious that $\xi_i \leq \xi$ for all $i \in I$. We next check that $\xi \in \Xi$. Let $s, t \in \text{dom}(\xi)$, such that $s \sim t$. Then $s \in \text{dom}(\xi_i)$ and $t \in \text{dom}(\xi_j)$ for some $i, j \in I$ and either $\xi_1 \leq \xi_2$ or $\xi_2 \leq \xi_1$. Assume $\xi_1 \leq \xi_2$. Then $s, t \in \text{dom}(\xi_j)$ and $\xi_j \in \Xi$ so we have that $\xi_j(s) \equiv_{\sim} \xi_j(t)$ showing that $\xi(s) \equiv_{\sim} \xi(t)$ and we have established that $\xi \in \Xi$.

Hence, every ascending chain in Ξ has an upper bound. By the Lemma of Zorn we conclude that Ξ has a maximal element. Let σ be such a maximal element in Ξ . We claim that σ is a total map. Assume opposite, i.e., there exists $s \in S \setminus \text{dom}(\sigma)$. If there exists a $t \in \text{dom}(\sigma)$ such that $s \sim t$ then we define a new partial scheduler σ' as follows. If $\sigma(t) = \perp$ we put $\sigma'(s) = \perp$. If $\sigma(t) = \mu_t$, then, since $t \rightarrow \mu_t$ and $s \sim t$, there exists μ_s such that $s \rightarrow \mu_s$ and $\mu_t \equiv_{\sim} \mu_s$. In this case we put $\sigma'(s) = \mu_s$. Moreover, put $\sigma'(x) = \sigma(x)$ for $x \in \text{dom}(\sigma)$. Then we have $\sigma' > \sigma$ and $\sigma' \in \Xi$ contradicting the maximality of σ . Hence σ is a total map.

Finally, we consider the (history-independent) scheduler $\hat{\sigma}$ induced by σ , i.e., defined by $\hat{\sigma}(\pi) = \sigma(\text{last}(\pi))$ for any finite path π . This scheduler is admissible. Namely, given π_1 and π_2 such that $\text{trace}(\pi_1) = \text{trace}(\pi_2)$ and $\text{last}(\pi_1) \sim \text{last}(\pi_2)$ we have, since $\sigma \in \Xi$, that

$$\hat{\sigma}(\pi_1) = \sigma(\text{last}(\pi_1)) \equiv_{\sim} \sigma(\text{last}(\pi_2)) = \hat{\sigma}(\pi_2)$$

which completes the proof. \square

We are now ready to define anonymity for probabilistic systems, the formal definition follows.

Definition 6.3 (Anonymity). A system $\langle M, I, \{A_i\}_{i \in I}, \text{Act}_O \rangle$ is said to be anonymous if for all admissible schedulers ξ , for all $i \in I$ and for all $o \in O$

$$\begin{aligned} \mathbb{P}_{\xi}[\pi \in A] > 0 \implies \mathbb{P}_{\xi}[\text{Otrace}(\pi) = o \wedge \pi \in A_i \mid \pi \in A] = \\ \mathbb{P}_{\xi}[\text{Otrace}(\pi) = o \mid \pi \in A] \mathbb{P}_{\xi}[\pi \in A_i \mid \pi \in A]. \end{aligned}$$

7 Anonymity Examples

In the purely non-deterministic setting, anonymity of a system is often proved (or defined) as follows: take two users A and B and a trace in which user A is “the culprit”. Now find a trace that looks the same to the adversary, but in which user B is “the culprit” [HO03,GHvRP05,MVdV04,HK07]. In fact, this new trace is often most easily obtained by switching the behavior of A and B .

In this section, we make this technique explicit for anonymity in our setting, with mixed probability and non-determinism.

Definition 7.1. Let M be a probabilistic automaton. A map $\alpha: S \rightarrow S$ is called an *Act_O-automorphism* if α induces an automorphism of the automaton M_τ , which is a copy of M with all actions not in Act_O renamed to τ .

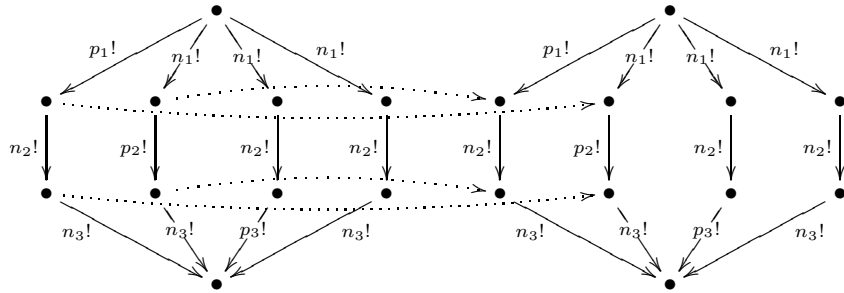
The following result generalized the above-mentioned proof technique that is commonly used for a purely non-deterministic setting.

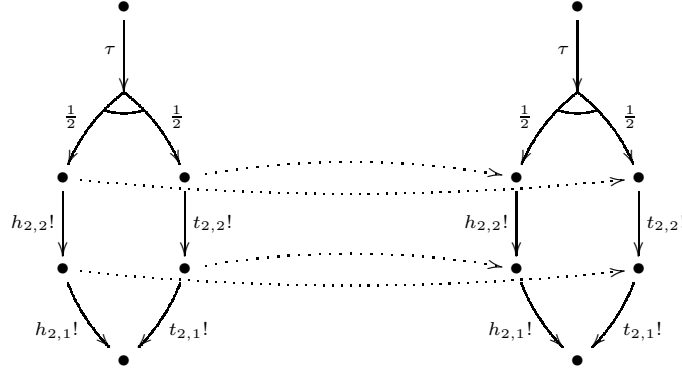
Theorem 7.2 *Consider an anonymity system (M, I, Act_O) . Suppose that for every $i, j \in I$ there exists a Act_O -automorphism $\alpha: S \rightarrow S$ such that $\alpha(A_i) = A_j$. Then the system is anonymous.*

Anonymity of the Dining Cryptographers

We can now apply the techniques from the previous section to the Dining Cryptographers. Concretely, we show that there exists a Act_O -automorphism exchanging the behaviour of the Crypt_1 and Crypt_2 ; by symmetry, the same holds for the other two combinations.

Consider the endomorphisms of Master and Coin_2 indicated in the following figure. The states in the left copy that are not explicitly mapped (by a dotted arrow) to a state in the right copy are mapped to themselves.





Also consider the identity endomorphism on Crypt_i (for $i = 0, 1, 2$) and on Coin_i (for $i = 0, 1$). Taking the product of these seven endomorphisms, we obtain an endomorphism α of DC.

References

- [Bel57] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.
- [BP05] Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In Martín Abadi and Luca de Alfaro, editors, *Concurrency Theory, 16th International Conference (CONCUR '05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 2005.
- [Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. 1(1):65–75, 1988.
- [Che06] Ling Cheung. *Reconciling nondeterministic and probabilistic choices*. PhD thesis, RU Nijmegen, 2006.
- [GHvRP05] Flavio D. Garcia, Ichiro Hasuo, Peter van Rossum, and Wolter Pieters. Provable anonymity. In Ralf Küsters and John Mitchell, editors, *Proceedings of the 2005 ACM Workshop on Formal Methods in Security Engineering (FMSE '05)*, pages 63–72. ACM, 2005.
- [HK07] Ichiro Hasuo and Yoshinobu Kawabe. Probabilistic anonymity via coalgebraic simulations. In *European Symposium on Programming (ESOP '07)*, volume 4421 of *Lecture Notes in Computer Science*, pages 379–394. Springer, Berlin, 2007.
- [HO03] Joseph Halpern and Kevin O'Neill. Anonymity and information hiding in multiagent systems. In *16th IEEE Computer Security Foundations Workshop (CSFW '03)*, pages 75–88, 2003.
- [MVdV04] S. Mauw, J. Verschuren, and E.P. de Vink. A formalization of anonymity and onion routing. In P. Samarati, P. Ryan, D. Gollmann, and R. Molva, editors, *Proceedings of Esorics 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 109–124, 2004.
- [Seg95] R. Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, MIT, 1995.
- [SL94] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In *Proc. Concur'94*, pages 481–496. LNCS 836, 1994.

- [Sok05] A. Sokolova. *Coalgebraic analysis of probabilistic systems*. PhD thesis, TU Eindhoven, 2005.
- [SV04] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, pages 1–43. LNCS 2925, 2004.